AMENDMENTS

Please amend the following claims by adding the language that is underlined and by deleting the language that is struck-through:

23- (Currently Amended) A method for passing information to a post-compile-time software application, comprising the steps of:

compiling a plurality of blocks of code~~;~~ , including finding one or more unused bits in an instruction in one of the plurality of blocks of code ~~that are compiled;~~ , and using the one or more unused bits to encode information ~~pass information to the post-compile-time software application~~ ~~.~~ ; and

communicating the information to the post-compile-time software application for use by the post-compile-time software application.

24- (Previously Added) The method of claim 23, wherein the information identifies whether certain registers are live.

25- (Previously Added) The method of claim 23, wherein the post-compile-time software application comprises a dynamic optimizer.

26- (Previously Added) The method of claim 23, wherein the instruction is a no-operation (NOP) instruction.

27- (Currently Amended) The method of claim 23, ~~further comprising: using the information~~ wherein the information is used by the post-compile-time software application to determine whether certain registers are live.

28- (Previously Added) The method of claim 23, wherein the information is encoded as a bit vector.

29- (Currently Amended) The method of claim 23, wherein the step of using the one or more unused bits to pass information to the post-compile-time software application comprises:

determining which of a plurality of registers are live in said one of the plurality of blocks of code;

creating within the instruction a register-usage bit-vector having a plurality of register-usage bits; and

setting one of the plurality of register-usage bits for each one of the plurality of registers that are live.

30- (Currently Amended) A system comprising:

a compiler that is configured to compile a plurality of blocks of code ; ~~and~~ , the compiler including a code annotator that is configured to find one or more unused bits in an instruction in one of the plurality of blocks of code that are being compiled by the compiler, and to encode information in the one or more unused bits ~~; wherein the information is~~, the information being configured to be used by a post-compile-time software application ~~.~~ ; and

a processor for executing the compiler.

31- (Previously Added) The system of claim 30, wherein the information identifies whether certain registers in said one of the plurality of blocks of code are live.

32- (Previously Added) The system of claim 30, wherein the post-compile-time software application comprises a dynamic optimizer.

33- (Previously Added) The system of claim 30, wherein the instruction is a no-operation (NOP) instruction.

34- (Previously Added) The system of claim 30, wherein the post-compile-time software application is configured to use the information to determine whether certain registers are live.

35- (Previously Added) The system of claim 30, wherein the information is encoded as a bit vector.

36- (Currently Amended) A system comprising:

means for compiling a plurality of blocks of code ~~;~~ , including finding one or more unused bits in an instruction in one of the plurality of blocks of code, and using the one or more unused bits to pass information to a post-compile-time software application; and

means for executing the means for compiling.

~~means for finding one or more unused bits in an instruction in one of the plurality of blocks of code that are compiled by the means for compiling; and~~

~~means for using the one or more unused bits to pass information to a post-compile-time software application.~~

37- (Previously Added) The system of claim 36, wherein the information identifies whether certain registers are live.

38- (Previously Added) The system of claim 36, wherein the post-compile-time software application comprises a dynamic optimizer.

39- (Previously Added) The system of claim 36, wherein the instruction is a no-operation (NOP) instruction.

40- (Currently Amended) A method for compiling, comprising the steps of:

~~compiling a plurality of blocks of code;~~

finding one or more unused bits in an instruction in one of a plurality of blocks of code that are being compiled; and

encoding information in the one or more unused bits ~~; and using the information~~ , wherein the information is used by a post-compile-time software application.

41- (Previously Added) The method of claim 40, wherein the information identifies whether certain registers are live.

42- (Previously Added) The method of claim 40, wherein the post-compile-time software application comprises a dynamic optimizer.

43- (Previously Added) The method of claim 40, wherein the instruction is a no-operation (NOP) instruction.

44- (Previously Added) The method of claim 40, further comprising:

using the information by the post-compile-time software application to determine whether certain registers are live.

45- (Previously Added) The method of claim 40, wherein the information is encoded as a bit vector.

46- (New) A method for passing information to a post-compile-time software application, comprising the steps of:

compiling a plurality of blocks of code; and

during the step of compiling, finding one or more unused bits in an instruction in one of the plurality of blocks of code, wherein the one or more unused bits are used to pass information to the post-compile-time software application.

47- (New) The method of claim 23, wherein the information identifies whether certain registers are live.

48- (New) The method of claim 23, wherein the instruction is a no-operation (NOP) instruction.